5.0 The Technology Environment

In compliance with the Information Technology Reform Management Act of 1996, the BLM is developing a Bureau Architecture. This will provide a mission-driven framework for BLM's Information Technology investment decisions. All of the technology decisions and investments for the NILS Project will be based on the Bureau Architecture. This includes designing and building an integrated structure of processes, information flows, software, and hardware to meet the business process requirements. The Forest Service and other NILS partners will make their investment decisions on their own internal architecture needs. The following subsections provide a high-level overview of the computing environment requirements.

5.1 Computing Environment

5.1.1 Desktop Environment

The client computing environment for NILS would be industry standard desktop platforms that provide a windowing graphical user interface.

5.1.2 Server Environment

The server computing environment for NILS would be industry standard platforms that can host object or relational database management systems and support a transactional, multi-user client base.

5.1.3 Intranet Environment

Applications outside of GeoCommunicator may be capable of performing database updates. They may connect to the central database(s) using secure Intranets and wide-area networks.

5.1.4 Internet Environment

GeoCommunicator may be hosted inside of web browsers. Most applications may run inside the browser itself, for data catalog browsing, mail, forums, and so on. GeoCommunicator facilities for data upload and download would be based on standard Internet protocols such as the File Transfer Protocol (FTP).

5.1.5 In-Field Computing Environment

The use cases in the Survey Management business process area would require interfacing with field instruments to transfer data between the instruments and the system. They also would require computational support for survey activities. Survey Management outputs then feed into the Measurement Management subsystem for the construction of the *legal description fabric.*

There is a spectrum of computing capabilities that could be provided to the field surveyor. The high end of the spectrum would provide full Measurement Management capabilities, allowing measurement network construction and editing to be performed in the field. Errors or inconsistencies in the results could be resurveyed before the crew leaves the project site.

The low-end solution would provide field computation support and survey device interfacing only, perhaps running on small portable computers.

5.2 Software

NILS would be built using commercial off-the-shelf (COTS) software products as much as possible. Customization for specific hardware and system interfaces would be needed, as well as extensions to COTS software for BLM-specific survey practices or other work processes. Developing web sites, and handling reports and map formats may require customization for the specific business needs of other NILS partners.

5.2.1 Object-Oriented Analysis, Design and Development Methodology

Overview of OOAD

An object-oriented analysis and design (OOAD) method was used to capture the essential business process requirements that would be supported by the NILS software application.

The object-oriented approach, as it relates to the design and development of software applications, focuses on modeling the real-world entities that are involved in an integrated set of business processes. Essential business processes are identified, named and described as a system's use cases. Use cases serve as conceptual containers for the series of steps that are performed to complete a given workflow process. As the description of the use case becomes more detailed, a set of real-world actors, inputs, documents, forms, processes, interactions and outputs are identified. These process-related entities are 'candidates' for the types of software 'objects' that must be designed and developed.

The OOAD approach was selected as the NILS analytical and design method for two important reasons:

- Modeling focuses on the actual business process requirements of the system users.
- The analysis and design methods are part of an industry standard, comprehensive method of software development that is cost-effective, flexible and maintainable.

Object Orientation (OO)

The object-oriented method contrasts with previous generations of software development that utilized procedural languages and thus produced 'monolithic' applications. Such software was characterized as being extremely customized and expensive both to develop and to maintain (it was not easy to modify as new functionality or environmental factors were introduced). In many cases users were forced into a workflow environment to accommodate the programming language rather than their desired business operations.

The object-oriented approach emerged as an improved development environment that broke the design-once, build-once 'monolithic' model by replacing procedures with a specification for component-to-component interfacing. That meant that specific parts of the system (component) could be built individually as separate software 'classes', as long as each class followed the protocol for requesting and providing processing services with other classes. Classes could be built to provide the system functionality required to process information based on the real-world entities (objects) that were involved. And as the nature or behavior of an object required modifications, the software could accommodate the programming update without a major re-writing of code. Object-oriented programming languages were created, and standard protocols for object-to-object interfacing were adopted.

Clients, Classes, Objects

These new software classes are templates for 'objects' that are created in a computer's processing memory area. Objects are system binary files that act as 'clients' (sending requests for services) and 'servers' (using the parameters in the request to complete a process and return information). An object-oriented software application uses an integrated set of objects that interact to retrieve, process and store information.

Objects have properties, behaviors and states. The state of an object can be thought of as the set of its current properties (variable and attribute data values). Behavior refers to the methods or operations that can be performed on an object (as defined by the specific class from which the object was 'instantiated'). It is because objects are 'encapsulated' (they have their own states and available behaviors) that object-oriented applications are so efficient and flexible to maintain.

For example, a parcel object would have inherent properties to manage its state and attributes (active/inactive, planned, historical, versioned, edited, etc.) A parcel object would have inherent behaviors that govern how it may be created, deleted (inactivated and stored in an historical repository), edited (split, merged, reshaped), adjusted (in a parcel fabric), linked to external data, and symbolized for display and plotting.

Component Object Model (COM)

Object classes can be independently developed as software components because standard protocols for object-to-object interfacing have been adopted. Microsoft created the Object-Linking and Embedding (*OLE*) and COM specifications and finally evolved ActiveX using COM. COM has become the desktop software industry standard for object-oriented development.

Unified Modeling Language

UML can be used to specify component interfaces and relationships. The use case method utilized in documenting the NILS business process requirements is a UML-supported approach.

Several Computer Aided Software Engineering (CASE) tools have been developed which can create UML diagrams (graphical representations of data models) and can engineer both database schemas and software code. The schema can be implemented in commercial relational database management systems. CASE tools can create code in the form of proto-objects (stub-code) already set up in a user-specified programming language and ready for additional programming to implement object behavior, rules and properties.

Why Geo Objects, COM and OO are Better

- Classes, as the building blocks for software applications, are based on ***user-specified*** object properties and behaviors needed to perform ***user-specified*** business operations.
- Object-components are based on standards-there are clearly-defined and well-understood standards that define component interfaces precisely.

- Extensibility-because components are based on standards, anyone (core developers, third party developers and end users) can create them and extend a software system. New components can be created by specializing existing components.
- Powerful design and analysis tools-there is a rich collection of CASE tools and specialized component suites.
- Language neutrality and independence-many software languages support COM, and applications can utilize COM-compliant classes developed in any language.
- Efficiency and cost-effectiveness of maintenance and change incorporation.
- Inter-process capability-supports distributed computing (e.g., on distributed systems and over the Internet).

NILS would be built using a use case driven object modeling process. The analysis and design would be documented using the notation of UML. Whenever possible, modeling objects would be mapped to or derived from software classes, reusing the functionality of COTS software.

5.2.2 COTS Products

The key features of a desktop software product that may be relevant to the NILS project are:

- an extensible geographic object model with customizable object behaviors and interactions,
- support for long transactions,
- support for survey measurement networks and networks of constructions (import, computation, cogo constructions, least squares),
- support for land parcel transactions and lineage,
- DBMS-based data storage,
- extensible through standard object-oriented technology.

In addition, the software may provide support for the mapping and reporting capabilities needed by the GeoCommunicator. It would be driven in part by the same back-end database technology as the rest of the system.

5.2.3 Customization Requirements

Field Instruments

Specific hardware and system interfaces for survey instruments and field computers may be needed for each adopting organization.

BLM-Specific Survey Practices

BLM-specific survey practices would probably require the incorporation of some GCDB Measurement Management (GMM) functionality.

Web Site Development

NILS would acquire COTS products for managing e-mail, forums, notices, events, and accounts. The design of the GeoCommunicator web site's pages and visuals, data upload and download procedures, and other interfaces would require development and customization.

Report and Map Format and Content

An extensive set of products may be produced. These would be based on both the core data models and customized parcel fabrics or data layers.

Base Parcel Fabric Customization

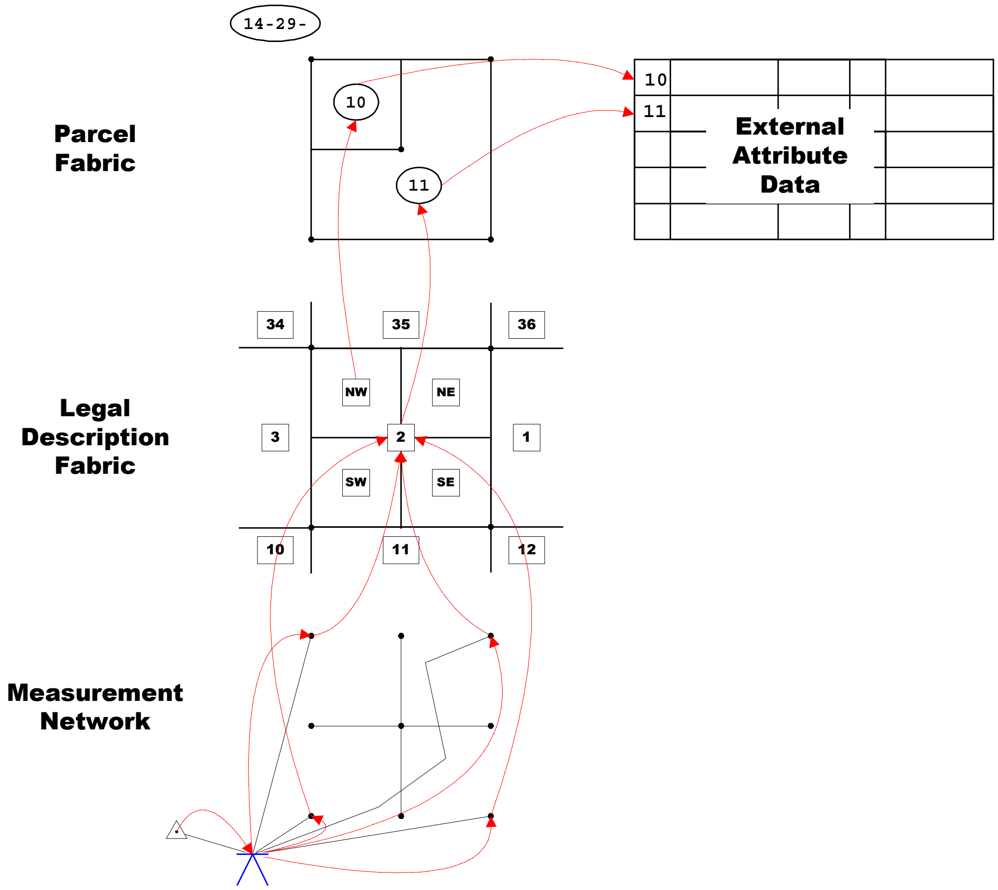
Each type of homogenous parcel fabric (public lands, assessment parcels, zoning, registry/ownership, etc.) may require a data model based on the common legal description fabric, tailored to the business and data needs of each adopting organization.

5.3 Data

NILS supports two broad categories of land records data: the map data delineating parcels, and the non-spatial record information about those parcels. Since there are major non-mapping-based data management systems in place in many jurisdictions where NILS would be applicable, NILS needs to be concerned primarily with the problems of maintaining the parcel map data. The map data would, of course, carry along keys to non-spatial record information so that record data can be maintained in conjunction with parcel data.

The NILS data architecture would be multi-tiered (Figure 5.1). Survey observations (e.g., traverses, measurements) define coordinate locations that serve as the basis for constructing measurement networks. Point elements in the measurement network are aggregated into measured features. The geometric elements (lines and areas) of the legal description fabric are constructed from the underlying measurement network elements. Parcel features in the parcel fabric are built from one or more areas or boundaries contained in the legal description fabric. Parcels are associated to external databases (e.g., assessment, ownership, case management).

Figure 5.1. Multi-tier Data Architecture

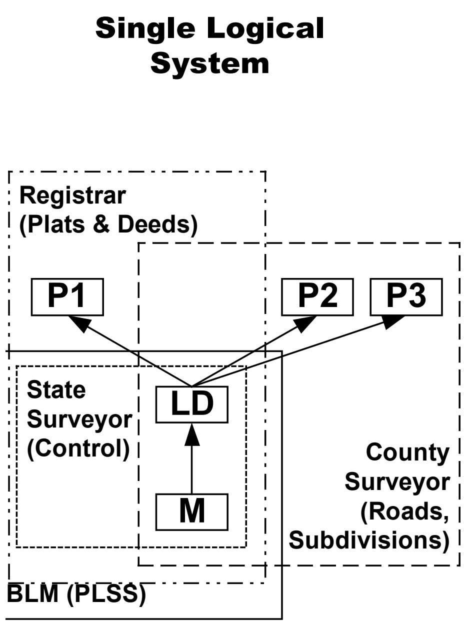


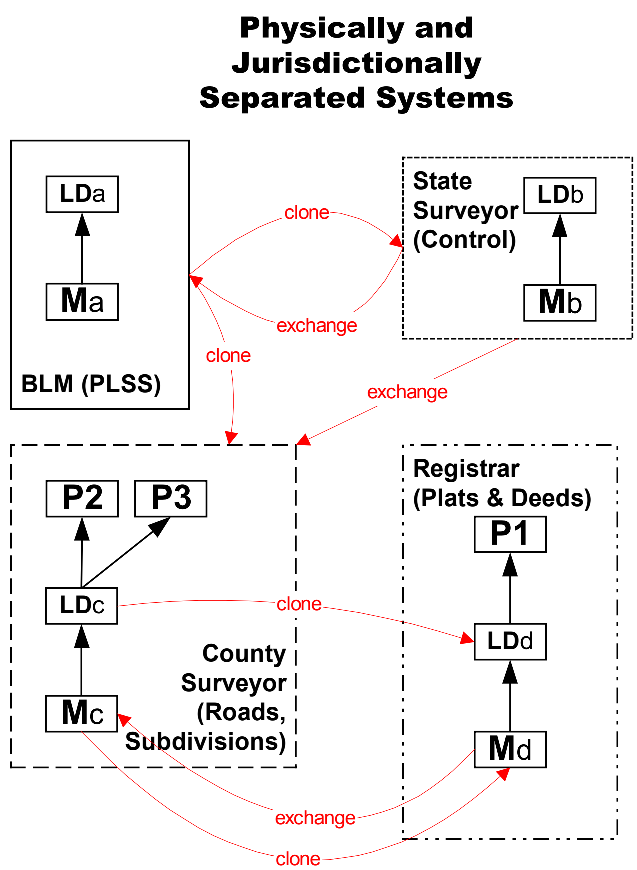
There are several new challenges for the data model defined by NILS. NILS assumes the ability to maintain multiple, complex relationships among all data elements described in this document. Specifically, NILS conceives of a set of interrelated, distributed data that can be shared and/or copied between different organizations, and selectively synchronized as shown in Figure 5.2.

Security and auditing is of prime importance. First and foremost, it keeps data secure. It is also the mechanism used to review mapping decisions made by one individual or an entire organization before incorporating data into a departmental system or another organization's system. It forms the basis of the selective exchange of information shown Figure 5.2.

Figure 5.2. Data Relationships among Organizations

KEY: M = Measurement Network; LD = Legal Description Fabric; P1…P3 = Parcel Fabrics. In a single system, database integrity is straightforward because parcel fabrics all point to a common legal description fabric. In multi-system configurations, versions (clones) must be managed when data is shared (exchanged). Version management includes tracking changes, conflict resolution and synchronization.





5.4 Concept for NILS Data Model

The relationships between NILS survey measurements, legal descriptions, parcels, external parcel attributes, and transactional information are shown in Figure 5.3. See Section 7.0 for more detail on general requirements.

Figure 5.3. Relationships among NILS Information